WHAT IS CLAIMED IS:

Sub D1 E1

1. A computerized wagering game apparatus, comprising:

a computerized game controller having a processor, memory, and nonvolatile storage and is operable to control the computerized wagering game; and

an operating system comprising: a system handler application operable to dynamically link with at least one gaming program object; and

an operating system kernel that executes the system handler application.

2. The computerized wagering game apparatus of claim 1, wherein the system handler application comprises a plurality of device handlers.

3. The computerized wagering game apparatus of claim 1, wherein the system handler application comprises software having the ability when executed to:

load a gaming program shared object; and

execute the new gaming program shared object.

Sub D2

4. The computerized wagering game apparatus of claim 1, wherein game data modified by the gaming program objects is stored in nonvolatile storage.

5. The computerized wagering game apparatus of claim 4, wherein changing game data in nonvolatile storage causes execution of a corresponding callback function in the system handler application.

Sub D3

6. The computerized wagering game apparatus of claim 1, wherein the computerized game controller comprises an IBM PC-compatible controller.

7. The computerized wagering game apparatus of claim 1, wherein the operating system kernel is a Linux operating system kernel.

8. The computerized wagering game apparatus of claim 7, wherein the Linux operating system kernel is modified.

9. The computerized wagering game apparatus of claim 8, wherein the kernel has at least one modification wherein each modification is selected from the group consisting of: 1) accessing user level code from ROM, 2) executing from ROM, 3) zero out unused RAM, 4) test and/or hash the kernel, and 5) disabling selected device handlers.

10. The computerized wagering game apparatus of claim 9, wherein the modifications are modular.

11. The computerized wagering game apparatus of claim 1, wherein the system handler application comprises an API with functions callable from the gaming program objects.

12. The computerized wagering game apparatus of claim 1, wherein the system handler further comprises an event queue.

Sub D 4/ 13. A method of managing data in a computerized wagering game apparatus via a system handler application, comprising:

loading a shared object,

executing the shared object, and

accessing and storing game data in nonvolatile storage.

14. The method of claim 13, and further comprising the step of unloading the first program object, and further comprising loading a second program object.

15. The method of claim 13, further comprising executing a corresponding callback function upon alteration of game data in nonvolatile storage.

16. A computerized wagering game system controlled by a general-purpose computer, comprising an operating system kernel that is customized for gaming use.

17. The computerized wagering game system of claim 16, wherein the kernel is customized in at least one way; selected from the group: 1) accessing user level code from ROM, 2) executing from ROM, 3) zero out unused RAM, 4) test and/or hash the kernel, and 5) disabling selected device handlers.

18. A computerized wagering game system controlled by a general-purpose computer comprising nonvolatile storage that stores game data, such that loss of power does not result in loss of the state of the computerized wagering game system.

19. A gaming machine operating system, comprising a processor and memory and is operable to control the computerized wagering game, wherein the memory contains a plurality of shared objects and a system handler, and the system handler is adapted to execute at least one shared object called from memory.

20. The operating system of claim 19, further comprising nonvolatile storage, wherein game data stored in nonvolatile storage is retained during a gaming machine power down.

21. A machine-readable medium with instructions thereon, the instructions when executed operable to cause a computer to:

load a first program shared object,

execute a first program shared object,

store game data in nonvolatile storage, such that a second program object later loaded can access the data variables in nonvolatile storage,

unload the first program shared object, and

load the second program shared object.

22. The machine-readable medium of claim 21, with further instructions operable when executed to cause a computer to execute a corresponding callback function upon alteration of game data in the nonvolatile storage.

23. The machine-readable medium of claim 22, with further instructions operable when executed to cause a computer to manage events via the system handler application.

24. A machine-readable medium with instructions thereon, the instructions when executed operable to cause a computer to manage at least one gaming program object via a system handler application, such that a single gaming program object is executed at any one time, wherein gaming program objects are operable to share game data in nonvolatile storage.

25. The machine-readable medium of claim 21, wherein only one gaming program object executes at any one time.

26. The machine-readable medium of claim 21, with further instructions operable when executed to cause a computer to provide functions through an API that comprises a part of the system handler application.

27. A machine-readable medium with instructions thereon, the instructions when executed are operable to store game data in nonvolatile storage, such that the state of the computerized wagering game system is maintained when the machine loses power.

28. A gaming machine architecture, comprising an operating system, and a plurality of shared objects; wherein each shared object describes game personality in a selected mode.

29. The gaming machine architecture of claim 28, wherein the operating system comprises an IBM PC-based operating system.

30. The gaming machine architecture of claim 28, wherein the operating system comprises a system handler.

31. The gaming machine architecture of claim 30, wherein the system handler comprises a plurality of device handlers.

32. The gaming machine architecture of claim 30, wherein the system handler comprises an event queue.

33. The gaming machine architecture of claim 30, wherein the system handler comprises a plurality of API callable functions.

34. The apparatus of claim 1, wherein the system handler comprises an event queue that determines the order of execution of each specified device handler.

35. The apparatus of claim 1, wherein the system handler comprises an API having a library of functions.

36. The apparatus of claim 34, wherein the event queue is capable of queuing on a first come, first serve basis.

37. The apparatus of claim 34, wherein the event queue is capable of queuing using more than one criteria.

38. The apparatus of claim 1, wherein the system handler and kernel work in communication to hash the system handler code and operating system kernel code.

39. A universal operating system, comprising a system handler; and an operating system kernel.

40. The operating system of claim 39, and further comprising a plurality of APIs.

41. The operating system of claim 39, and further comprising an event queue.

42. The operating system of claim 39, wherein the system handler comprises a plurality of device handlers.

43. The operating system of claim 39, wherein the operating system kernel is customized for gaming purposes.

44. The operating system of claim 43, wherein the kernel is customized utilizing a method selected from the group consisting of: 1) Accessing user level code from ROM, 2) executing from ROM, 3) zero out unused RAM, 4) test and/or hash the kernel, and 5) disabling selected device handlers.

45. The operating system of claim 39, wherein the system is used to control a networked on-line system.

46. The operating system of claim 39, wherein the system is used to control a progressive meter.

Sc5 D7 47. A method of modifying an operating system kernel, comprising at least one modification to obtain functionality selected from the group consisting of:

1) accessing user level code from ROM;
2) executing user level code from ROM;
3) zeroing out unused RAM;
4) testing and/or hashing the kernel; and
5) disabling selected device handlers.

Add A1

Add B1

Add D7